

Git and GitHub for Beginners

Every software team uses Git. As a tester you will store and share your automation code with it.

1. What are Git and GitHub?

- **Git** is a tool installed on your computer. It saves every version of your code, so you can go back if something breaks, and many people can work on the same code without overwriting each other.
- **GitHub** is a website that stores Git projects online (similar sites: GitLab, Bitbucket). It adds sharing, review and teamwork on top of Git.

Simple idea: Git is like "save history" in a game, and GitHub is the cloud where the team keeps the shared save.

2. Words you must know

TERM	MEANING
Repository (repo)	A project folder tracked by Git.
Commit	A saved snapshot of your changes, with a message.
Branch	A separate line of work. You can change code safely without touching the main code.
main (master)	The main, stable branch.
Remote / origin	The online copy of the repo (on GitHub). "origin" is its default nickname.
Clone	Download a full copy of a remote repo to your computer.
Fork	Your own personal copy of someone else's repo on GitHub.
Pull request (PR)	A request to merge your branch into main. Teammates review it first.
Merge	Combining one branch's changes into another.

Merge conflict	Two people changed the same lines; Git cannot decide, so you choose manually.
Staging area	Waiting room where you place changes before committing.

3. How the work flows



And in the other direction, `git pull` brings the latest changes from GitHub to your computer.

4. First-time setup

```
git config --global user.name "Your Name"
git config --global user.email "you@example.com"
```

What it does: tells Git who you are. This name and email appear on every commit you make. Do it once per computer.

5. Commands you will use daily

COMMAND	WHAT IT DOES
<code>git clone <url></code>	Download a repo from GitHub.
<code>git status</code>	Show which files changed. Use it often.
<code>git add file.java</code>	Put one file in staging. <code>git add .</code> adds all changed files.
<code>git commit -m "message"</code>	Save staged changes with a message.
<code>git push</code>	Upload your commits to GitHub.
<code>git pull</code>	Download and merge the latest changes from GitHub.
<code>git log --oneLine</code>	Show the commit history in short form.
<code>git diff</code>	Show exactly what lines you changed.

`git fetch`

Download remote changes but do not merge yet.

Branch commands

COMMAND	WHAT IT DOES
<code>git branch</code>	List branches.
<code>git checkout -b feature/login-tests</code>	Create a new branch and move to it. (<code>git switch -c</code> does the same.)
<code>git checkout main</code>	Move to an existing branch. (<code>git switch main</code> also works.)
<code>git merge feature/login-tests</code>	Merge that branch into the branch you are on.
<code>git branch -d feature/login-tests</code>	Delete a branch you no longer need.

Undo and save-for-later commands

COMMAND	WHAT IT DOES
<code>git restore file.java</code>	Throw away your unsaved changes in that file.
<code>git restore --staged file.java</code>	Remove a file from staging, keep your changes.
<code>git commit --amend</code>	Fix the last commit message or add a forgotten file (only before pushing).
<code>git revert <commit-id></code>	Safely undo an old commit by making a new opposite commit.
<code>git stash</code> / <code>git stash pop</code>	Temporarily keep unfinished work aside, then bring it back.

Be careful with `git reset --hard`. It deletes your changes permanently. Ask a senior before using it.

6. A normal working day

```
git checkout main
git pull
git checkout -b feature/add-search-tests
# ... write your test code ...
git status
git add .
git commit -m "Add search page automation tests"
git push -u origin feature/add-search-tests
```

Step by step: go to main → get the latest code → make your own branch → work → check what changed → stage the files → commit with a clear message → push the branch to GitHub (`-u` links your local branch to the remote one, needed only the first time). Then open GitHub and create a **pull request**.

7. Pull request (PR) process

1. Push your branch and click "Compare & pull request" on GitHub.
 2. Write a short title and description of what you changed.
 3. Teammates review and may leave comments. You fix them and push again; the PR updates itself.
 4. After approval (and passing automatic checks), the PR is merged into main.
-

8. Merge conflict

When Git cannot merge automatically, it marks the file like this:

```
<<<<<<< HEAD
String url = "https://qa.example.com";
=====
String url = "https://staging.example.com";
>>>>>>> feature/add-search-tests
```

How to fix: the top part is your current branch, the bottom part is the incoming branch. Keep the right line (or combine both), delete the `<<<<<<<`, `=====`, `>>>>>>>` lines, then run `git add` and `git commit`.

9. Good habits

- **Never** work directly on main. Always create a branch.
- **Pull before you start** work, to avoid conflicts.
- Commit small and often with clear messages: "Add login test for invalid password", not "changes" or "fix".
- Use a `.gitignore` file to keep unwanted files out of Git, such as `target/`, `.idea/`, logs, and screenshots.
- **Never commit passwords, tokens or API keys.** Once pushed, they are visible to everyone with access.

10. How testers use Git

- Automation code (Selenium, Rest Assured) lives in a repo, just like developer code.
- Your test scripts go through branches and pull requests.
- CI tools (Jenkins, GitHub Actions) pull the repo and run your tests automatically on each change.
- Testers also check a bug's linked commit or PR to see what the developer changed, so they know what to retest.

11. More commands and ideas

Starting a project from scratch

```
git init
git remote add origin https://github.com/yourname/qa-project.git
git branch -M main
git push -u origin main
```

What it does: `init` makes the folder a Git repo → connects it to an empty GitHub repo called origin → names the branch main → uploads the first commits.

Three kinds of reset

COMMAND	COMMIT	STAGING	YOUR FILES
<code>git reset --soft HEAD~1</code>	Undone	Kept	Kept

<code>git reset --mixed HEAD~1</code>	Undone	Cleared	Kept
<code>git reset --hard HEAD~1</code>	Undone	Cleared	Lost

`HEAD` means the commit you are on now. `HEAD~1` means one commit before it. Use reset only on commits you have **not pushed**. For pushed commits use `git revert`.

Other useful commands

COMMAND	WHAT IT DOES
<code>git remote -v</code>	Show the remote URLs.
<code>git cherry-pick <commit-id></code>	Copy one particular commit onto your current branch.
<code>git tag v1.0</code>	Mark a commit with a name, such as a release.
<code>git blame file.java</code>	Show who changed each line and when.
<code>git show <commit-id></code>	Show the details of one commit.
<code>git rebase main</code>	Move your branch's commits on top of the latest main.
<code>git branch -a</code>	List local and remote branches.
<code>git log --oneline --graph</code>	History with a branch drawing.

Merge vs rebase

- **Merge:** joins two branches with a merge commit. Keeps full history. Safe and simple.
- **Rebase:** replays your commits on top of another branch for a straight, clean history.
Never rebase branches that others are using.

.gitignore example

```
target/
.idea/
*.log
screenshots/
config/secrets.properties
```

What it does: Git will skip the build folder, IDE settings, log files, screenshots and a secrets file. Files already committed must be removed with `git rm --cached` first.

HTTPS vs SSH for GitHub

HTTPS asks for a username and a personal access token. **SSH** uses a key pair saved once on your computer, so you do not type anything again. Use `ssh-keygen` to create the key and add the public key to GitHub settings.

12. Branching strategy in companies

BRANCH	PURPOSE
main	Stable, ready-for-release code.
develop	Where the team's finished features come together for testing.
feature/xyz	One new feature or one set of tests.
bugfix/xyz	A fix for a bug.
hotfix/xyz	A quick fix straight from main for a live problem.
release/1.2	Final stabilizing before a version is released.

13. GitHub features you will see

- **Issues:** track tasks and bugs (like Jira tickets).
- **Pull request reviews:** reviewers can comment, request changes or approve. Be polite and clear in comments.
- **README.md:** the first page of the project. Explain what it is and how to run the tests.
- **Branch protection:** main cannot be pushed to directly; a PR with approval and passing checks is needed.
- **GitHub Actions:** runs your tests automatically when code is pushed.

```
name: Run tests
on: [push, pull_request]
jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-java@v4
        with: { java-version: '17', distribution: 'temurin' }
      - run: mvn test
```

What it does: on every push or pull request, GitHub starts a fresh machine → downloads your code → sets up Java 17 → runs `mvn test`. If a test fails, the PR shows a red cross. This is basic CI (Continuous Integration). Jenkins does the same job on your own server.

CI vs CD

CI (Continuous Integration): code is built and tested automatically on every change. **CD (Continuous Delivery/Deployment):** after tests pass, the app is released to an environment automatically or with one click.

14. Quick interview answers

- **Git vs GitHub?** Git is the version control tool on your computer. GitHub is the online platform to host and share Git repos.
- **git pull vs git fetch?** `fetch` only downloads changes. `pull` = fetch + merge.
- **git merge vs git rebase?** Merge joins branches and keeps history as it happened. Rebase moves your commits on top of the latest main to make a straight history. (Fresher level: knowing merge is enough.)
- **What is a branch for?** To work on a feature or fix without disturbing the stable code.
- **What is git stash?** Temporarily saves unfinished changes so you can switch branches with a clean folder.
- **How do you solve a merge conflict?** Open the file, decide which lines to keep, remove the markers, add, and commit.
- **What is a pull request?** A request for teammates to review your changes before they are merged.

- **How do you undo the last commit but keep the changes?** `git reset --soft HEAD~1`.
- **git init vs git clone?** `init` makes a new empty repo. `clone` copies an existing repo from a server.
- **Revert vs reset?** Reset moves history back (use for local commits). Revert adds a new commit that undoes an old one (safe for pushed commits).
- **What is HEAD?** A pointer to the commit or branch you are working on.
- **What is cherry-pick?** Copying one specific commit to another branch.
- **Fork vs clone?** Fork makes your own copy on GitHub. Clone downloads a repo to your computer.
- **What is .gitignore?** A file listing what Git should not track.
- **Staging area?** The place where you collect changes before committing.
- **What is CI/CD?** Automatic build, test and release on every change. Tools: Jenkins, GitHub Actions.
- **How did you use Git as a tester?** Cloned the framework, made a feature branch, added tests, raised a PR, fixed review comments, merged.